#9

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No.	: 09/603,356	Confirmation No.	3257
Applicant	: Cheng, Ray et al		
Filed	: 06/26/2000		
TC/A.U.	: 2153		
Examiner	: EDELMAN, BRADLEY E.		
Docket No.	: 77666-5		
Customer No.	: 07380		

DECLARATION UNDER 37 CFR 1.131

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

I, Allan Brett of 177 Cameron Street, Ottawa, Ontario, Canada, K1S 0X4 make oath and say:

1. I am a patent agent with the firm of Smart & Biggar, and have been involved in the preparation and prosecution of the above-noted application.

2. Attached as Exhibit "H" to this Declaration is a photocopy of a facsimile received on February 17, 2000 from the inventor Ray Cheng. The facsimile provides information for the preparation of the patent application for the present invention. Please note that Exhibit "H" refers to the same document as Exhibit "A" to Ray Cheng's Declaration Under 37 CFR 1.131, but the body of the facsimile is also enclosed, this showing the details of the invention as contemplated at that time.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that wilful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such wilful false statements may jeopardize the validity of the application or any patent issued thereon.

Respectfully submitted

By _____
Allan Brett

Dated: Mar. 18, 2004
Ottawa, Ontario, Canada

EXHIBIT H

Visit our Web site at: www.entrust.com

Phone: 232-2486

Facsimile: 232-8440

Date: Feb 17, 2000

To: ALAN BRETT

Company: **Pages to Follow:** 9

From: RAY CHENG 247-3174

Message

Document on Multi-Domain Single Sign-On - a design and one page describing broader concepts

[redacted]

Entrust Technologies

Multi-Domain Single Sign-On

Author: Ray Cheng
Date: January 25, 2000
Version: 0 2

Revision Table

Issue	Date	Summary
0.1	January 19, 2000	First draft
0.2	January 25, 2000	Added support for custom page and servlet settings

1 Problem of multi-domain single sign-on (MDSSO)

1.1 The problem and the solution

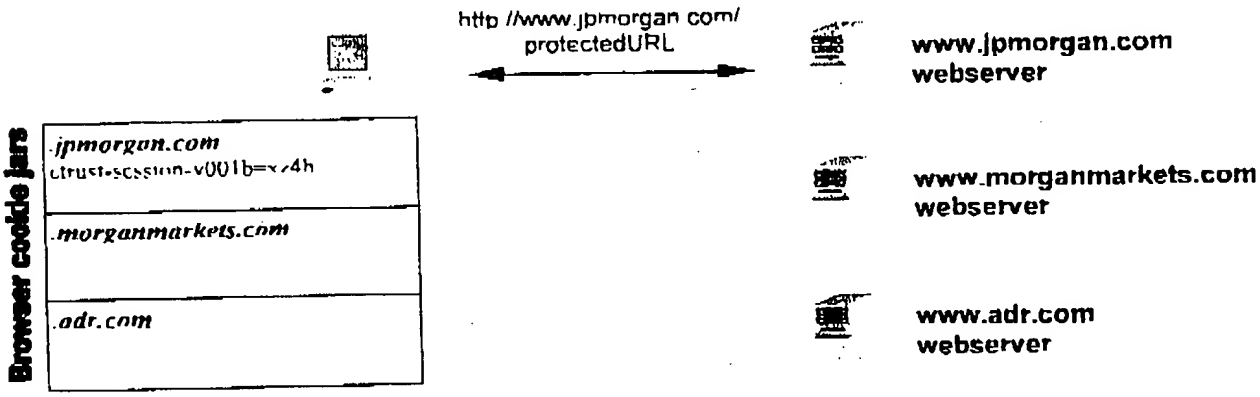
Solving the MDSSO problem is really about bypassing the security in the cookie handling mechanism which prevents one site/domain from reading or modifying cookies that are used by another site/domain

The solution needs to bypass enough of the cookie security to allow domains – in a family of trusted domains – to copy cookies generated by one domain into the cookie jars of the other domains at the browser

Since the solution is about sharing cookies among multiple domains, we should keep the product which solves the MDSSO problem as general as possible and not introduce any dependencies on SecureControl or SSO.

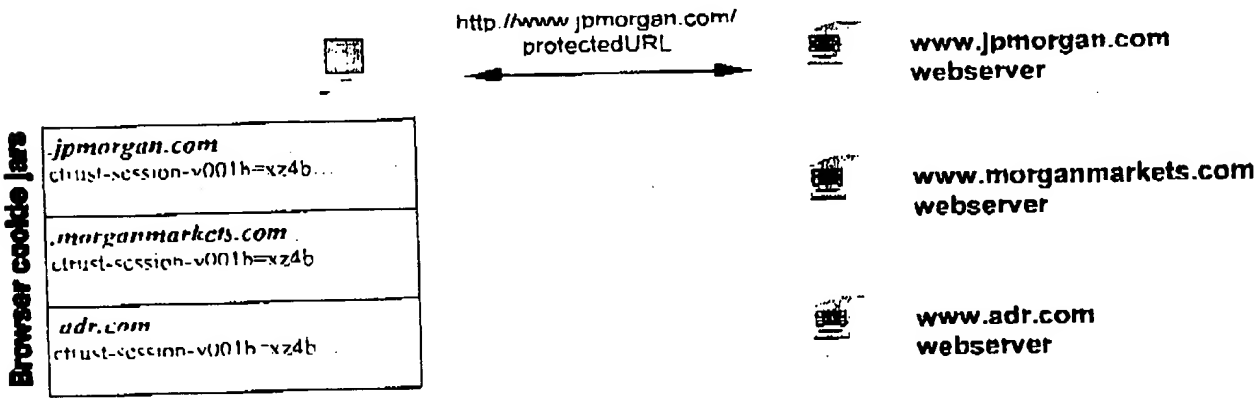
1.2 Baking cookies

The following shows what happens to a browser's cookie jars when he visits a SecureControl protected URL at www.jpmorgan.com and is successfully authenticated and authorized:



The user next visits a protected URL on either www.morganmarkets.com or www.adr.com He will need to re-authenticate since the SecureControl SSO cookie that he got from www.jpmorgan.com will not be sent to either one of these sites

To achieve MDSSO, we want all his cookie jars to be filled with the SecureControl SSO cookie after his initial visit to www.jpmorgan.com.



If we do this, when he next visits a protected URL on www.morganmarkets.com or www.adr.com, he will NOT need to re-authenticate since a valid SecureControl SSO cookie would be sent along with his request.

2 Putting cookies into forbidden jars

Some methods which may be employed to write cookies from one domain into the cookie jars of other domains

1. **Enable data tainting at the client and the webserver return a client-side script which writes the SSO cookie into the cookie jar of the other domains.** Unfortunately, this disables cookie security for all sites and works only for some versions of Netscape Navigator.

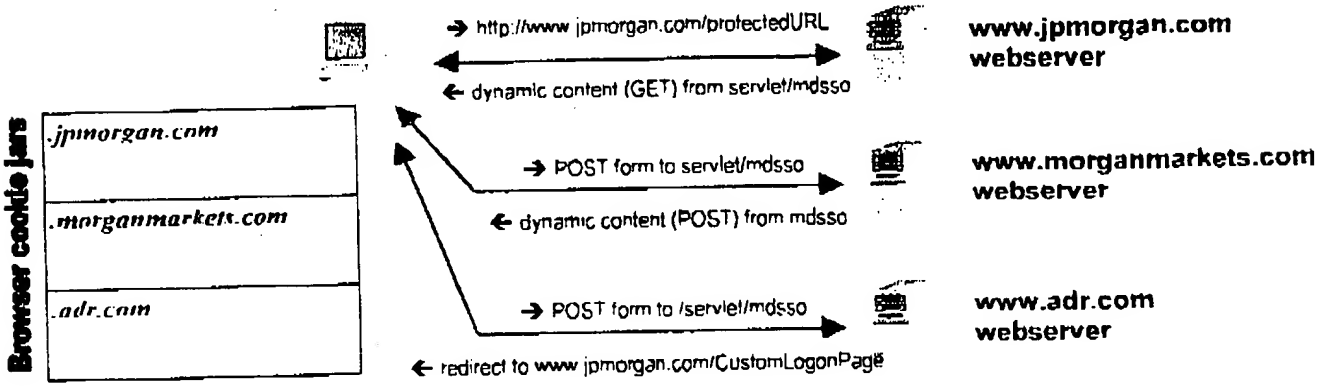
2. **Webserver sends a signed client-side script which writes the SSO cookie into the cookie jar of the other domains.** This solution probably causes more problems than it solves since there is no consistent standard for handling signed scripts among browsers and some browser versions do not handle signed scripts at all. The actual signing of scripts and maintenance of keys and certificates is another major administration issue

3. **After receiving a SSO cookie, have the client visit all the other domains in the MDSSO family and get the other domains to echo back the SSO cookie into their corresponding cookie jar at the browser.** This is the only practical solution since it requires only basic features of the HTTP and uses web content that is understood by most browsers. The rest of this document will examine the mechanics of achieving this *cookie routing using client-side Javascript*[1] in more details.

[1] Although it is simpler to populate the various cookie jars with redirection and pass the necessary information along in the parameter portion of the URL, inconsistent limits imposed by webservers and browsers on the maximum size of the URL parameters make this impractical

3 Cookie routing using client side Javascripts

3.1 *Step-by-step walk through*



1. User requests protected URL for the first time on www.jpmorgan.com.
2. SecureControl plugin detects that the URL is protected and redirects the user to the form authentication page.
3. User submits his username/password via the form
4. SecureControl plugin authenticates user. If authenticated and authorized, redirects user to the successful logon page (e.g. /servlet/mdsso). In this case, the successful logon page is a servlet[2] which performs the MDSSO.

What does the EnableMDSSO servlet do?

a. Check for a MDSSO cookie

The MDSSO cookie may contain the following information. The data that goes in the cookie is a bit up in the air right now. *It may not be necessary to have this cookie at all depending which configuration setting we want at each webserver*

- time of last MDSSO – this is to used to ensure that MDSSO is refreshed only after a configurable period has passed since the last MDSSO and to keep us from going in circles passing cookies around
- domains in the MDSSO family that have been refreshed – this may not be necessary depending on how we want to configure the different webservers. For example, if we store at each webserver the *next* site in the domain family rather than the list of sites in the domain family, then we may not need to store this information in the MDSSO cookie

[2] In this walk through, servlets are used as the vehicle for generating dynamic content. These servlets can be replaced by other means of dynamic content generation such as CGIs, ASPs and JSPs. A bug in the ASP engine which incorrectly encode cookie names and contents prevents it from being used however

b. Generate a client-side javascript that is run when the HTML <BODY> is loaded by the client

```
<head>
<script language "javascript">
function doMDSSO()
        document.forms[0].submit();
}
 /script>
</head>

<body language="javascript" onLoad="doMDSSO()">
```

c Embed a hidden form with the hidden values.

```
<form method="post" action="http://www.morganmarkets.com/servlet/mdsso"
  name="mdsso">
  <input type="hidden" name="etssoc" value="xz4b ">
  <input type="hidden" name="homeServer" value="www.jpmorgan.com">
  <input type="hidden" name="homeURL" value="/CustomLogonPage" >
  <input type="hidden" name="mdsso" value="www.jpmorgan.com" >
</form>
```

- The http://www.morganmarkets.com/servlet/mdsso value is derived from the serlvet properties at the webserver. This is the URL that the form data will be *posted* to
- The value for "etssoc" is the SecureControl SSO cookie extracted from the request header.
- The value for "homeServer" and "homeURL" tells us which server the MDSSO process began and the page that we want to end up at when the MDSSO process is over. These are servlet properties set at each MDSSO server.
- The value for "mdsso" is a list of domain which have been visited in this case – what we want to put is this cookie is not yet final at this point

5 Browser receives dynamic content generated by /servlet/mdsso (GET). It runs the function doMDSSO() and post the form data to the next server in the MDSSO family (http://www.morganmarkets.com/servlet/mdsso in this example).

6. The mdsso servlet at www.morganmarkets.com compares:
   a the server that the client will post to next
   b and the homeServer in the post data

   Since the two values are different, the mdsso servlet generates the dynamic content similar to the mdsso servlet at www.jpmorgan.com. The exception is that the next client side post will be to http://www.adr.com/servlet/mdsso.

7 The mdsso servlet at www.adr.com compares
   c the server that the client will post to next
   d and the homeServer in the post data

   Since the two values are the same, it sends a redirection header back to to client and the client sees the CustomLogonPage at the server that was initially visited

4 mdsso servlet properties

Some example properties that needs to be configured at each webserver in the MDSSO family.

	webserver		
Property	**www.jpmorgan.com**	**www.morganmarkets.com**	**www.adr.com**
mdsso.next server	www morganmarkets com	www.adr com	www.jpmorgan com
mdsso.next uri	/servlet/mdsso	/servlet/mdsso	/servlet/mdsso
mdsso home.server	www jpmorgan com	www morganmarkets.com	www.adr com
mdsso.home uri	/CustomLogonPage	/CustomLogonPage	/CustomLogonPage

We may also have some settings (i.e. name, path, expiry and domain) for the MDSSO cookie should we decide to use one.

Thursday, February 17, 2000

Broader concepts for patenting the MDSSO technology

One way to view the invention is as a method to transfer data across a series of communications devices (e.g. across a series of servers, without any direct server-server communication)

Another way to view the invention is as a method to transfer data from two or more servers into a browser (e.g. the SSOC from the first server).

Concept 1 (novel use of browser as "data delivery agent")

- using browser as an agent to deliver data (identical, or modified) from one server to another, with no direct server-server communications. For example, : browser, server1, browser, server2, , browser, server_n (Optionally finish up at the browser also.)
- differs from the conventional use of a browser (as master) acquiring data from a server. Also conventionally data acquired is not automatically forwarded to another server. Prior art includes a browser "posting" data to a server, e.g. when a user fills out a form, but then the browser delivers data the user has entered (vs. data received from another server).
- either done transparently to end-user, or by presenting a choice to browser user re: whether to allow use of browser for slave action. The latter may be better depending on the application/data and issues (i.e. privacy, feeling of being in control from a consumer perspective)
- specific case of HTTP: data can be transferred as a parameter in a GET, or content in a POST
-

•Concept 2 (specifically, using communications protocol like HTTP to distribute SSOC)

- Approach 1: pre-store information at the servers, specifying a "next server" in a virtual list. (Perhaps more secure, but harder to update the list and/or add server components.)
- Approach 2a: information sent to the server includes an ordered list (e.g. [a,b,c,d]) of servers to visit. A given server finds itself in the list, and causes browser to visit next server in list (sending list along).
- Approach 2b: optimize Approach 2a by sending on only the un-used tail end of the list.

•Note: 2a and 2b seem better than 1 if the servers are not maintained by one company (see the write-once-shop-anywhere example). No harm in mentioning both in the patent application.

3?

-

•Concept 3 (more broadly, converting master-slave to slave-master)

- method to turn master-slave protocol into slave-master, without changing the protocol. For example, novel use of existing HTTP protocol so browser becomes slave to the server and follows server commands (beyond simple prior-art use of re-direct)
- other examples: two-way-pager and network base station, handheld communications device and network server.
- consider a PalmPilot or RIM two-way pager being used to transmit data from server1 to server2, where privacy laws preclude server1 from directly communicating to server2; this might also be necessary for some strange legal reasons, e.g. for historical laws to hold in cyberspace a direct contractual relationship may need to be established between end-user and each site

EXHIBIT A

Visit our Web site at: www.entrust.com

Phone: 232-2486

Facsimile: 232-8440

Date: Feb 17, 2000

To: ALAN BRETT

Company: Pages to Follow: 9

From: RAY CHENG 247-3174

Message

Document on Multi-Domain Single Sign-On architecture and one page describing broader concepts.